Journal of Computational Science 15 (2016) 3–10

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Workflow optimization of performance and quality of service for bioinformatics application in high performance computing

Rashid Al-Ali, Nagarajan Kathiresan*, Mohammed El Anbari, Eric R. Schendel, Tariq Abu Zaid

Biomedical Informatics Research Division, Sidra Medical and Research Center, Doha 26999, Qatar

ARTICLE INFO

Article history:
Received 14 July 2015
Received in revised form 21 February 2016
Accepted 4 March 2016
Available online 24 March 2016

Keywords:
High performance computing
BWA-MEM algorithm
Quality of service
Next generation sequencing
Scalability
Application performance and parallel efficiency

ABSTRACT

Nowadays, High Performance Computing (HPC) systems commonly used in bioinformatics, such as genome sequencing, incorporate multi-processor architectures. Typically, most bioinformatics applications are multi-threaded and dominated by memory-intensive operations, which are not designed to take full advantage of these HPC capabilities. Therefore, the application end-user is responsible for optimizing the application performance and improving scalability with various performance engineering concepts. Additionally, most of the HPC systems are operated in a multi-user (or multi-job) environment; thus, Quality of Service (QoS) methods are essential for balancing between application performance, scalability and system utilization. We propose a QoS workflow that optimizes the balancing ratio between parallel efficiency and system utilization. Accordingly, our proposed optimization workflow will advise the end user of a selection criteria to apply toward resources and options for a given application and HPC system architecture. For example, the BWA-MEM algorithm is a popular and modern algorithm for aligning human genome sequences. We conducted various case studies on BWA-MEM using our optimization workflow, and as a result compared to a state-of-the-art baseline, the application performance is improved up to 67%, scalability extended up to 200%, parallel efficiency improved up to 39% and overall system utilization increased up to 38%.

1. High performance computing for bioinformatics

Due to various advancements in next-generation sequencing technologies (e.g. Illumina, SOLiD), larger volumes of genome data are being produced every year at a lower cost [1]. New functional variants are being discovered due to this ever-growing availability of genome data [2]. However, the analysis applications required for these discoveries typically are performance limited due to their compute and memory-intensive operations [3]. This paper addresses these challenges by optimizing genome alignment applications that are commonly hindered when using traditional High Performance Computing (HPC) systems. To overcome the traditional system limitations, HPC systems are becoming popular in bioinformatics for providing faster genome alignment by utilizing high-throughput and parallel-processing techniques [4,5], referred as "HPC for Bioinformatics" [6]. Thus, large-scale genome analysis can be parallelized to achieve empirically faster results by using HPC architectures, but those gains still have much room for improvement.

Nowadays, multi-core HPC systems used in genome sequencing still have no optimal choice of workflows based on application characteristics, in terms of accuracy, performance and optimal selection of computing resources. Generally, the application performance is dependent on various factors like complexity of the algorithm, application design, data distribution methods, communication cost, workflow dependency conditions, software stack (e.g. compilers, Message passing Interface (MPI)/Thread libraries) and hardware limitations [7]. To achieve the optimal performance of any application, it is necessary to understand the application characteristics and the performance bottlenecks.

Most bioinformatics applications are written in multi-threaded programming models that do not scale well in the modern multi-core HPC systems [3,8]. For example, when a modern GATK Haplotype caller application [9] is executed on a 32-cores HPC system with core steps of 2, 4, 8, 16 and 32, the performance improvement was expected as execution time keep reducing with the increased number of cores. However, when beyond 8 cores, the performance was not improving in relation. Therefore, the optimal computing resources should be selected based on the scalability

* Corresponding author.
E-mail address: nkathiresan@sidra.org (N. Kathiresan).

http://dx.doi.org/10.1016/j.jocs.2016.03.005

limitations to achieve the better performance. Alternatively, the utilization of the HPC system is very poor (about 25% of resource utilization) for the above GATK Haplotype caller workload. In this case, it is necessary to improve the system utilization by concurrently submitting more similar workloads within the HPC system. We conducted various case studies of different ways of parallelization and their performance impacts are summarized in [8]. As of now, tools are unavailable to balance between application performance, scalability and system utilization and hence we introduced the Quality of Service (QoS) factor, which uses an effective ratio between parallel efficiency and system utilization. We designed this QoS as an optimization workflow to provide the best performance and linear scalability with optimal set of resources.

2. Literature review

In the last few years, hash table based algorithms [10] (e.g. BLAST, SOAP, SeqMap, etc.) and prefix/suffix trees based algorithms (e.g. FM-Index, BWA-MEM, BWT-SW) are commonly used in genome mapping in bioinformatics research [1]. Burrows-Wheeler Aligner (BWA) is the most popular genome mapping software widely used in human genomic sequencing [11–13]. The BWA-backtrack, BWA-SW and BWA-MEM are three different algorithm versions of BWA. The BWA-SW and BWA-MEM algorithms are supported for long-reads (70 bp–1 Mbp) human genome sequences. Unlike the other algorithms, the BWA-MEM provides fast and accurate alignment for sequence reads and support for long-query and split-alignment in the human genome sequencing [14].

BWA-MEM, BWA-BT, Bowtie2, SMALT and MOSAIC are some of the widely used aligner tools. The aligned reads in BWA-MEM and SMALT are greater than 99%, where the execution time of SMALT is 3 times slower than BWA-MEM. The BWA-MEM and Bowtie2 execution times are relatively comparable to each other but, the Bowtie2 aligned reads are relatively good (98.27%) compared to BWA-MEM (99.10%) [15].

A new MICA aligner is optimized to take advantage of Intel's Many Integrated Core Architecture (MIC), which is 4.9 times faster in execution time compared to the BWA-MEM algorithm [16]. The Regional Hashing-based Alignment Tool (rHAT) produces accurate aligned reads, correctly aligned bases and excellent execution time [17]. In this paper, we compared the rHAT algorithm, even though it uses Hash-Indexing, in order to understand the computational limitations of the BWA-MEM. Overall, the new implementations of aligners, MICA and rHAT, are compared to BWA-MEM [16,17]. The aligned reads and aligned bases are comparatively similar to each other, but the BWA-MEM algorithm failed to produce better execution time due to CPU limitations. Hence, we are optimizing the BWA-MEM algorithm using "data-parallel and concurrent parallelization" [3].

The implementations [15–17] discussed prior are focused in reducing execution time and not the optimal selection of the computational resources. The utilization of the resources is equally important in a multi-user environment, and the performance is not always ideal when all the computing resources are utilized [8]. To address these challenges, we proposed an optimal workflow for bioinformatics applications that will give a better suggestion to balance between application performance, scalability and system utilization, referred to as "best QoS".

3. Workflow optimization for bioinformatics applications

We present a systematic sequence of approaches called "workflow optimization" for the bioinformatics applications on the HPC system. The workflow is developed based on experience and various performance engineering concepts. When the application source code is available, compiler optimization techniques are used to improve the application performance [18]. We used 4 sets of compiler optimization flags: default optimization (-O3/-O2 flag), vectorization, Single Instruction Multiple Data (SIMD) based tunings and architecture aware optimizations (e.g. AVX, AVX2, -qarch=pwr8). For every change in the compiler flags, a different versions of application binary is created and run with a subset of genome data to measure the application execution time, HPC system efficiency and resource utilization. The best set of results is referred as "un-optimized" and it is a "baseline" reference for our workflow optimization.

The application profile based analysis is used to optimize the licensed applications because of its pre-compiled binaries. By using the flat profile (e.g. using GNU profiler and Intel Vtune), the performance bottleneck of both types (source code and licensed based) of applications are analyzed [19]. Based on the flat profile results, the relationship between application instructions and low-level characterizations (e.g. cache miss, translation lookaside buffer (TLB) miss, etc.) are studied. Accordingly, the application is tuned (e.g. parallelize the instruction set, change the order of execution of the instruction to take benefit of the cached registered entries, etc.) and optimized to make use of low-level hardware features. Additionally, the genome data is equally partitioned into independent chunks and equal to the number of cores in the HPC system. The optimized binary, which is used as the "baseline" reference, is concurrently executed with independent chunks of genome data and then measurements are taken of the application execution time (last concurrent job completion time), HPC system efficiency and resource utilization. This set of performance number are referred as "concurrent parallelization" [3].

Due to the larger volume of genome data, the cache miss/translation lookaside buffer (TLB) misses are possible in genome alignment. Hence, the genome data is partitioned into independent multiple chunks (not necessarily equal to the number of cores) based on the level of cache misses. The optimized binary is executed in a multi-threading mode with every independent chunk of data. During binary execution, the number selection of multi-threads is determined for providing the best scalability factor. Accordingly, the system utilization is calculated. The overall execution time is sum of all the execution time of independent chunks of data processing time and this method is referred as "data-parallelization".

Most of the multi-threaded applications are affected by shared memory contentions. As a result, scalability limitations and poor system utilization are observed. To address these challenges, "data-parallel with concurrent parallelization" is introduced [3]. In this method, optimal number of cores is selected based on the scalability limitation using the data-parallelization concept. The genome data is independently partitioned into an optimal number of cores. The optimal binary is executed with independent partition of data concurrently across all and multi-threading is used, which is equal to the number of optimal number of cores. Additionally, hyper threading (HT) or simultaneous multi-threading (SMT) enabled options are studied to bring the best performance improvement when the application is not affected with shared memory contention.

Fig. 1 provides a workflow performance optimization overview of bioinformatics applications represented by step-by-step flowchart model. Additionally, we summarized our method of optimization in the automated scripting (Algorithm 1), which is described as follows:

Notations and assumptions:

1. The HPC system $C = \{C_1, C_2 \ldots C_n\}$ has 'n' cores.
2. The genome data $D = \{D_1, D_2 \ldots D_m\}$ can be partitioned into 'm' independent chunks.

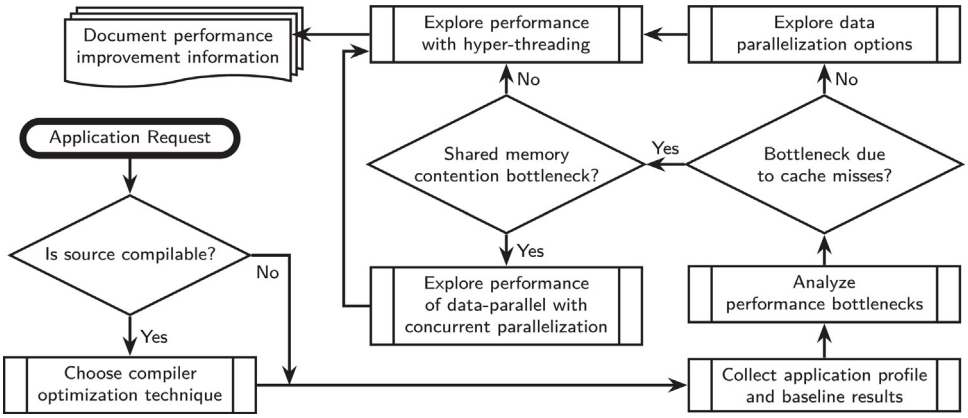

Fig. 1. Optimization workflow for bioinformatics applications.

3. The execution time $E = \{E_1, E_2 \ldots E_n\}$ with $E_1 > E_2 > \ldots > E_n$ is the set of application performance when the genome data D is processed in C multi-cores respectively.
4. The Scalability $S = \{S_1, S_2 \ldots S_n\}$ with $S_1 \leq S_2 \leq \ldots \leq S_n$ is the scalable performance when the application run with $\forall\, i = 1, 2, \ldots n$, C_i number of multi-cores respectively.
5. The efficiency $\eta = \{\eta/1, \eta/2 \ldots 1\}$ are the set of parallel efficiency when $\forall i = n, n-1, \ldots 1$ and C_i number of cores used respectively.
6. The best execution time, optimal scalability and better efficiency are referred as E_{best}, S_{opt} and η_{eff} respectively.

Algorithm 1. Method of proposed workflow optimization in the automated scripting.

Method: our proposed *workflow-optimization* is described as follows

input: Execute the application *A* to process the genome data D using C multi-core.
output: Provide the best combination of QoS(E_{best}, S_{opt} and η_{eff})
If A is source_code then
  For each compiler_flag(default, -O3, vectorization, SIMD, architecture aware ...) do
    Compile the source code;
     for any small subset $D_1 \in D$: Run A with D_1 using C_n;
    Measure the Execution_time=E_n;
    If $E_{best} > E_n$ then $E_{best} = E_n$ End if;
  End for; *// best compiler optimization is selected*
End if;
$\forall_{i=1,2\,..\,n}$, Run A with D using C_i and measure Scalability (S_i)
Select the S_{opt}=Optimal (S_i), η_{eff}=efficient(η_i), E_{best}=best(E_i); *// base line results obtained*
Generate the Flat profile;
Generate low level application profile data;
  Explore concurrent parallelization: Split data D into equal chunks $\forall_{i=1,2\,..\,n=m}$, D_i
  $\forall_{i=1,2\,..\,n=m}$: Run A with D_i using $C_{n/m}$ concurrently;
  Measure the over all execution time=max{E_i},.
  Select the S_{opt}=Optimal (S_i), η_{eff}=efficient(η_i), E_{best}= max{E_i}
    // best concurrent parallelization results
If (performance_bottleneck = Cache miss || TLB miss) then
  If (performance_bottleneck = shared memory contention) then
    Explore data-parallel with concurrent parallelization:
      $\forall_{i=1,2\,..\,n\,(n_{\neq}m)}$, Run A with C_i and D_i concurrently;
      Measure the over all execution time=max{E_i},.
      Select the S_{opt}=Optimal (S_i), η_{eff}=efficient(η_i), E_{best}= max{E_i}
        // best data-parallel with concurrent parallelization results
  End if
  Explore data parallelization: Split data D into $D_{n/x}$ partitions.
  $\forall$ $D_{n/x}$ partition: Run A with $D_{n/x}$ partition using C_n cores one by one.
  Measure the over all execution time=sum{ $\forall$ $D_{n/x}$ partition execution time}
  Select the S_{opt}=Optimal(S_i), η_{eff}=efficient(η_i), E_{best}= sum{ $\forall$ $D_{n/x}$}
    // best data- parallelization results
End if
Repeat the above steps with hyper threading enabled;
Report the best combination of QoS(E_{best}, S_{opt} and η_{eff});

4. Methodology

A. **Benchmarking environment:** due to advancement in cutting-edge technologies, a larger volume of genome sequences is being produced in high demand at lower cost using Next Generation Sequencing (NGS). The Burrows-Wheeler Transform (BWT) is a well-known permutation algorithm used in NGS [10,11,13]. Due to various enhancement in the in the NGS technology, the genome data size has keep increasing year by year and hence the higher demand of processing these data in a much faster way becomes necessary. The HPC systems addresses the data processing in an empirical parallel way and minimizes the overall execution time; hence, bioinformaticians prefer to process genome data on the HPC systems [12]. For our benchmarking exercise, we used the following two different high-end HPC systems:

- *Intel Sandy Bridge:* Four E5-4650 CPUs @ 2.7 GHz with hyper-threading enabled, 8 physical cores/CPU, totaling 64 cores.
- *AMD Opteron:* Four 6386 SE CPUs @ 2.8 GHz, 16 cores/CPU, totaling 64 cores.

Both the above HPC systems use Non-uniform memory access (NUMA) architectures and uses 512 GB DDR3 main memory. These high-end systems are running with Red Hat Enterprise Linux operating system, GNU compiler 4.4.7 version and BWA-MEM 0.7.10 version and hence the performance is comparable across the different HPC systems with the similar workload.

B. **Application selection:** most of bioinformatics applications are multi-threaded and available in open source implementations. The BWA-MEM algorithm is the most popular algorithm for aligning human genome sequences and it performs local alignment using BWT. This algorithm produces multiple primary alignments from different parts of query sequences that are widely used in the NGS technology [12]. The BWA-MEM alignment algorithm is written in C/Multi-threads (pThreads) implementation and it's processes the genome data using thread-parallelization, by default. Since the genome data has multi-million independent reads, the end-user can parallelize the data into independent chunks and process the genome alignment either simultaneously or one-chunk followed by-another. In the former case, all chunks will be processed by the HPC system at the same instant by overlapping is referred as "concurrent parallelization". Alternatively, in the later case, every chunk of data is processed one-by-one dedicatedly by the HPC system, which is referred as "data parallelization". Here, the most popular Message Passing Interface (MPI) based distributed processing or fault tolerance [7] (or multiprocessing across the nodes) is not possible because the BWA-MEM algorithm was not written with C/MPI. Based on our literature review [3,8,15–17], the BWA-MEM algorithm has failed to produce better execution time due to CPU limitations. Hence, we preferred to improve the performance by using our workflow optimization method.

C. **Genome data selection:** the Genome Comparison & Analytics Testing (GCAT) has standard set of genome data (100–400 bp) [20]. We used 100–150 bp paired-end data sets with large INDELs for our case studies because of our Illumina sequencer will produce a similar 100–150 bp human genome data.

D. **Results validation:** the output of the BWA-MEM algorithm is a SAM/BAM (Sequence Alignment/Map) file that contains TAB-delimited text consisting of a header section (optional) and an alignment section. The SAM file is a sister version of BAM that is in binary format. We used BamUtil tools [21] to validate the correctness of the generated SAM/BAM files. Even though the genome data are split into multiple independent chunks, the percentage of Mapping Rate, Paired Reads, Proper Pair, Duplicate Rate and QC Fail Rate should be same as "baseline" results. Additionally, the summation of number of read records and number of valid records between multiple chunks of genome data should be equal to the "baseline" results.

E. **Workflow optimization for BWA-MEM:** in thread parallelization, the BWA-MEM algorithm is used to run with 1, 2, 4,. . .N/4, N/2, N threads with the provided genome data file(s). These multi-threads (N) processes (e.g. mapping of genome sequence to the reference file) use the data file that contains multi-million reads for using functional parallelization. The data boundaries (e.g. reads line 1 to N/8 for thread=1, reads line N/8 + 1 to N/4 for threads = 2, etc.) are controlled by the way of implementation of sequence mapping algorithm. Generally, these algorithms are limited to performance bottlenecks due to cache/TLB misses and shared memory contention due to advancement in the multi-core era [10]. To eliminate those memory bottlenecks, the multi-million reads are equally partitioned into multiple chunks to reduce the read size.

Then the BWA-MEM algorithm is executed with multiple-threads within every chunk. All the chunk results are gathered into the final data parallelization resultant. This data parallelization uses the workflow model because of N threads are used to process every chunk of data and M times required to run BWA-MEM algorithm to process M number of chunks of data.

Alternatively, the concurrent parallelization is the best technique used to improve the throughput of multiple data files (e.g. multiple sequence data files) within pre-defined number of resources (e.g. N threads) with optimal execution time (e.g. less than $M\times$ of thread parallelization). This is explained as follows: the throughput in sequence alignment is referred as "multiple data files should be processed optimally with minimal execution time and limited number of resources" [8]. For example, 2 data files should be processed less than $2\times$ times of thread parallelization execution time with subject to N threads, i.e., first data file will be processed using N/2 threads and the second data file will be processed using another N/2 threads running concurrently. The maximum execution time of first and second data file should be less than $2\times$ time of thread parallelization execution time. Hence, the concurrent parallelization will be better for larger number of data files because of the optimal throughput performance.

F. **Performance measurement parameters:** the following definitions [4] are common in parallel processing and we used these performance metrics for comparing our workflow optimization benchmarking results:

- *Application performance:* measured as a BWA-MEM algorithm execution time E (in seconds).
- *Scalability:* the scalability is measured as a relative performance $S(p) = T_s/T_p$, where T_s is sequential execution time and T_p is parallel execution time of BWA-MEM algorithm. In an ideal case, $S(p) = p$, where p is the number of parallel threads used by BWA-MEM algorithm.
- *Quality of Service (QoS):* the best result of parallel efficiency is achieved by (i) tuning the application performance by using compiler optimization, architecture aware optimizations and application profile based tuning, (ii) the scalability of the application is improved by using data parallelization, concurrent parallelization and combination of both. The system utilization is calculated based on maximum utilized resources in the HPC architecture (e.g. number of CPUs). As a result, the proposed workflow optimization techniques help to reduce the computational and memory bottlenecks, minimize HPC resource idle time and various synchronization overhead, improve processor cache utilization for aligning concurrent data-intensive workloads and increases Quality of Service (QoS), which is calculated as:

(a) Efficiency: $\eta(p) = S(p)/p$ is parallel efficiency that measures relative performance of scalability versus number of

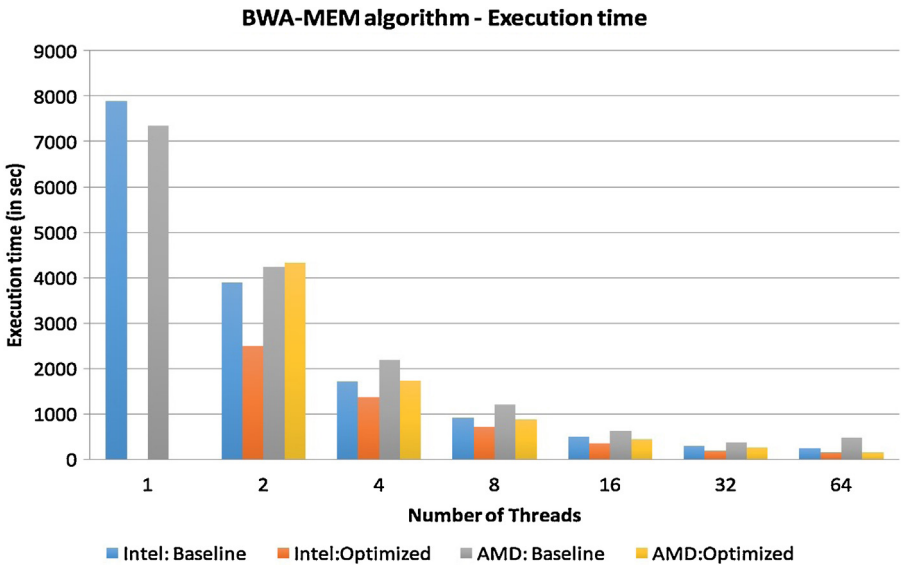

Fig. 2. Application performance of BWA-MEM algorithm.

parallel threads used by BWA-MEM algorithm. In an ideal case $\eta(p) = 1$ referred as QoS_{idle}.

(b) Resource utilization: $\delta(p) = (\eta(p) - QoS_{idle}) \times 100$, where $\delta(p) = 0$ is for 100% resource utilization, $\delta(p) < 0$ for under-utilization and $\delta(p) > 0$ for overutilization of HPC system resources. Due to various performance tunings (e.g. hybrid programming model) [3,8,10], their may be a possibility of improving the application performance super linearly and hence the resource utilization is demonstrated as a over utilization when the $\delta(p) > 0$.

5. Results and discussion

We followed our systematic multi-step approach to optimize the BWA-MEM application. The compiler optimization doesn't strongly influence the application performance improvement because it is dominated by indexing and sorting of the query genome sequences. The flat profile results also demonstrated the similar performance bottleneck with the increased number of cores. For example: subroutine ksw_u8() is taking >18–25% of total execution time with the increased number of cores in the parallel operation [3]. Additionally, we ran this BWA-MEM application on our high-end HPC systems (32 and 64 cores per node) and the performance keeps improving until cores = 16. With the larger core counts (32/64 cores), the performance is almost similar and these set of results are referred as "baseline" results or "un-optimized" performance numbers. The complete application performance results (cores = 1, 2, 4, ... 64) are summarized in Fig. 2.

We used "Amdahl's law of scalability" [22] to measure our "un-optimized" and "optimized" scalability results. In Intel system, "un-optimized" scalability results are super linear until cores = 8 and reduced scalability for cores = 32 and 64. On the AMD system, the "un-optimized" scalability results are reducing starting from cores = 4 and very poor for cores = 64. The summary of application scalability is illustrated in the Fig. 3

To improve the performance and scalability, the BWA-MEM algorithm is run with data parallelization and concurrent parallelization. The concurrent execution of multiple genome data (or independent chunks of single genome data) within a node

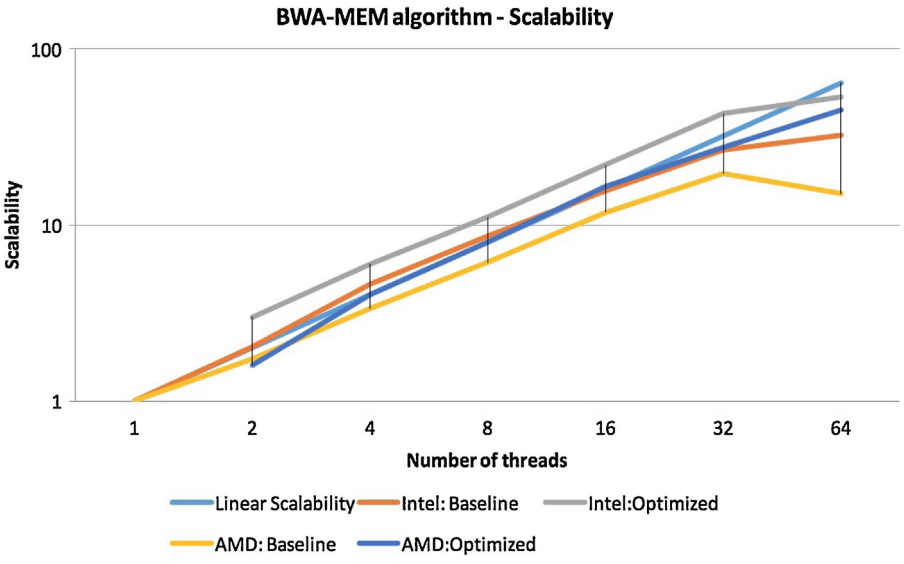

Fig. 3. Application scalability of BWA-MEM algorithm.

Table 1
Summary of QoS (parallel efficiency and system utilization).

# Thread	Intel: parallel efficiency		AMD: parallel efficiency		Intel: system utilization		AMD: system utilization	
	Un-optimized	Optimized	Un-optimized	Optimized	Un-optimized	Optimized	Un-optimized	Optimized
1	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
2	1.02	1.58	0.86	0.85	1.53	58.45	−13.53	−15.34
4	1.15	1.45	0.84	1.05	14.72	44.74	−16.42	5.49
8	1.08	1.38	0.76	1.05	7.88	38.06	−23.75	4.59
16	0.98	1.37	0.74	1.03	−2.46	37.10	−26.27	3.29
32	0.84	1.34	0.61	0.86	−16.35	34.12	−38.85	−14.12
64	0.51	0.83	0.24	0.70	−49.43	−16.63	−76.46	−29.66

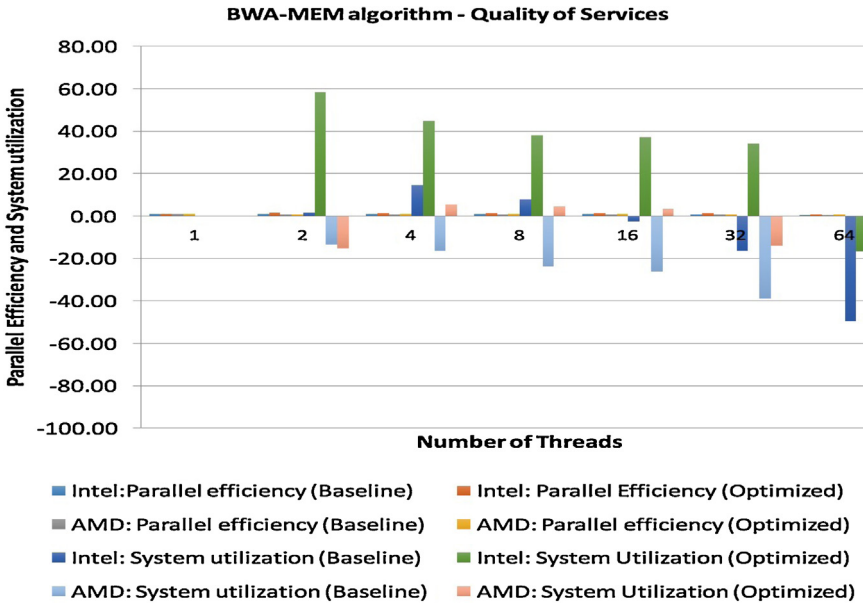

Fig. 4. Quality of service evaluation: parallel efficiency and system utilization.

provides the best throughput results and we suggested to run the BWA-MEM application using "concurrent parallelization" [8]. Alternatively, based on the best scalable number of cores, the genome data is partitioned into multiple chunks and run with data-parallel with concurrent parallelization using high-end HPC systems. As a result, the performance of BWA-MEM in Intel and AMD HPC systems are improved until cores = 64. The Intel scalability numbers are super-linear from cores = 2 to cores = 32 and AMD scalability is linear up to cores = 32. This set of improved results is referred as "optimized", which uses "data-parallel with concurrent parallelization" concept. The complete summary of optimized performance and extended scalability results are shown in Figs. 2 and 3.

In the above case studies, the efficiency of the HPC systems and resource utilizations are calculated based on the number of cores used. The Table 1 summarizes the parallel efficiency and system utilization based on our above stated performance measurement calculations, which is referred as Quality of Service (QoS).

As an overall outcome of workflow optimization, the BWA-MEM application performance is improved up to 39%, the scalability is extended up to 66%, parallel efficiency is improved up to 28% and overall system utilization is increased up to 38% on the Intel Sandy Bridge system. Similarly, the same set of benchmarks was performed on the AMD Opteron system, where the performance is impressive except for overall system utilization. As a summary of QoS, up to 67% application performance improvement is seen, up to 200% for extending scalability, up to 39% for improvement on parallel efficiency and up to 3% for overall system utilization that are illustrated in Fig. 4.

Results validation: the output of the BWA-MEM algorithm will produce the aligned genome data in SAM/BAM format. The BamUtil tool was executed to verify the correctness of the generated SAM/BAM files. The output of the BamUtil results (for cores = 1, 2, . . . 64) are the same across all the "baseline" results. The output of BamUtil is summarized in Fig. 5 as a reference.

When the genome data is partitioned into multiple independent chunks, the number of records read may not be similar across different chunks. Alternatively, we observed the percentage of Mapping

```
$ bam validate --in ./aln.64t.sam

Number of records read = 7963577
Number of valid records = 7963577

TotalReads(e6)   7.96
MappedReads(e6)  7.88
PairedReads(e6)  7.96
ProperPair(e6)   7.88
DuplicateReads(e6)       0.00
QCFailureReads(e6)       0.00

MappingRate(%)   98.94
PairedReads(%)   100.00
ProperPair(%)    98.94
DupRate(%)       0.00
QCFailRate(%)    0.00

TotalBases(e6)   1194.53
BasesInMappedReads(e6)   1181.84
Returning: 0 (SUCCESS)
```

Fig. 5. Output of BamUtil to verify correctness.

Table 2
Summary of BanUtil output in the multiple independent chunks.

	Chunk 1	Chunk 2	Chunk 3	Chunk 4	Sum (Chunk 1–4)
Number of records read	19,908,89	19,908,97	19,908,91	19,909,00	79,635,77
Number of valid records	19,908,89	19,908,97	19,908,91	19,909,00	79,635,77
Total reads (e6)	1.99	1.99	1.99	1.99	7.96
Mapped reads (e6)	1.97	1.97	1.97	1.97	7.88
Paired reads (e6)	1.99	1.99	1.99	1.99	7.96
Proper pair (e6)	1.97	1.97	1.97	1.97	7.88
Total bases (e6)	298.63	298.63	298.63	298.63	1194.52
Bases in mapped reads (e6)	295.44	295.44	295.49	295.49	1181.86

Rate, Paired Reads, Proper Pair, Duplicate Rate and QC Fail Rate are same across multiple independent chunks and importantly same as "un-optimized" results. To ensure the originality of records read, all the partial chunks records read are added together and the summary is available in Table 2.

From the above results, the summation of chunk-1 to chunk-4 results is matched with "baseline" results. In some of the cases, we observed, that the total bases (Fig. 5) and Bases in Mapped Reads values (Table 2) varies with 0.01–0.03% compared to our "baseline" results. Hence, our proposed workflow optimization provides 99.97% reliable and accurate results.

6. Future work

As of now, we are following the manual process of systematic workflow for optimizing the applications. This manual process requires time overhead to get the best results of QoS, which can be eventually automated by using continuous performance optimization techniques [23]. Additionally, we are in the process of including findings of all best combination in the parallel efficiency and system utilization (i.e., Best QoS) into our automated scripts. Therefore, whenever the application is rerun on the HPC system, the automated script will allocate the required resources based on our older observations (performance and resource); thus, the user application is automatically run with best parallelization and run time parameter options. Also, we are in the process of validating this optimization method using other standard bioinformatics alignment applications like Bowtie2, BLAST and SOAP and variant caller applications like GATK and RNASeq.

7. Conclusion

The proposed optimization workflow helps to improve the bioinformatics application performance and we have demonstrated this improvement in performance, scalability, system utilization and parallel efficiency using the most popular BWA-MEM alignment algorithm. We used a standard genome data downloaded from Bio-planet for our experiments on high-end HPC systems with two different architectures (Intel and AMD) up to 64 cores/node. As a result, the, performance was improved respectively on Intel and AMD systems by 39% and 67%, scalability extended to 66% and 200%, parallel efficiency improved by 28% and 39% in the high-end HPC systems. We were able to improve the system utilization up to 38% on Intel architecture. The optimized results are validated using the standard BamUtil and we observed 99.97% accurate results compared to baseline results. As a summary, the best QoS can be obtained using our proposed systematic workflow optimization for the bioinformatics applications.

Acknowledgements

The authors would like to thank Ramzi Temanni, Hakeem Almabrazi, Najeeb Syed and other Sidra Bioinformatics researchers for providing helpful comments and suggestion for running the BWA-MEM application.

References

[1] H. Li, N. Homer, A survey of sequence alignment algorithms for next-generation sequencing, Brief. Bioinform. 11 (5) (2010) 473–483.
[2] D.R. Bentley, Whole-genome re-sequencing, Current opinion in genetics & development 16 (6) (2006) 545–552.
[3] N. Kathiresan, M.R. Temanni, R. Al-Ali, Performance improvement of BWA MEM algorithm using data-parallel with concurrent parallelization, in: International Conference on Parallel, Distributed and Grid Computing (PDGC) 2014, IEEE, 2014, pp. 406–411.
[4] G. Hager, G. Wellein, Introduction to High Performance Computing for Scientists and Engineers, CRC Press, 2010.
[5] J. Zhang, H. Lin, P. Balaji, W.-C. Feng, Optimizing burrows-wheeler transform-based sequence alignment on multicore architectures, in: 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), 2013, IEEE, 2013, pp. 377–384.
[6] B. Schmidt, Bioinformatics, High Performance Parallel Computer Architectures, CRC Press, 2010.
[7] N. Gopalan, K. Nagarajan, Self-refined fault tolerance in HPC using dynamic dependent process groups, in: Distributed Computing-IWDC 2005, Springer, 2005, pp. 153–158.
[8] N. Kathiresan, R. Al-Ali, P.V. Jithesh, T. AbuZaid, R. Temanni, A. Ptitsyn, Optimization of data-intensive next generation sequencing in high performance computing, in: 15th International Conference on Bioinformatics and Bioengineering (BIBE), 2015, IEEE, 2015, pp. 1–6.
[9] A. McKenna, M. Hanna, E. Banks, A. Sivachenko, K. Cibulskis, A. Kernytsky, K. Garimella, D. Altshuler, S. Gabriel, M. Daly, et al., The genome analysis toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data, Genome Res. 20 (9) (2010) 1297–1303.
[10] W. Wang, W. Tang, L. Li, G. Tan, P. Zhang, N. Sun, Investigating memory optimization of hash-index for next generation sequencing on multi-core architecture, in: Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International, IEEE, 2012, pp. 665–674.
[11] M. Burrows, D. Wheeler, A block-sorting lossless data compression algorithm, in: Digital SRC Research Report, Citeseer, 1994.
[12] H. Li, Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM, 2013, arXiv preprint arXiv:1303.3997.
[13] E.R. Schendel, Y. Jin, N. Shah, J. Chen, C.-S. Chang, S.-H. Ku, S. Ethier, S. Klasky, R. Latham, R. Ross, et al., Isobar preconditioner for effective and high-throughput lossless data compression, in: IEEE 28th International Conference on Data Engineering (ICDE), 2012, IEEE, 2012, pp. 138–149.
[14] H. Li, R. Durbin, Fast and accurate short read alignment with burrows-wheeler transform, Bioinformatics 25 (14) (2009) 1754–1760.
[15] M. Mielczarek, J. Szyda, B. Guldbrandtsen, Optimizing NGS data analysis pipelines-comparison of alignment and variant calling tools, in: European Association for Animal Production, Annual Meeting, Book of Abstracts, 2014.
[16] S.-H. Chan, J. Cheung, E. Wu, H. Wang, C.-M. Liu, X. Zhu, S. Peng, R. Luo, T.-W. Lam, Mica, A fast short-read aligner that takes full advantage of intel many integrated core architecture (MIC), BMC Bioinform. 16 (2015), arXiv preprint arXiv:1402.4876.
[17] B. Liu, D. Guan, M. Teng, Y. Wang, rHAT: fast alignment of noisy long reads with regional hashing, Bioinformatics (2015), pii:btv662.
[18] D.A. Padua, M.J. Wolfe, Advanced compiler optimizations for supercomputers, Commun. ACM 29 (12) (1986) 1184–1201.
[19] J. Fenlason, R. Stallman, Gnu gprof, GNU Binutils, available at: http://www.gnu.org/software/binutils.
[20] Genome, Comparison and analysis testing: standard genome data, 2014 http://www.bioplanet.com/gcat.
[21] C. Genetics, BamUtil repositor perform operations on SAM and BAM files, 2014 http://genome.sph.umich.edu/wiki/BamUti.
[22] M.D. Hill, M.R. Marty, Amdahl's law in the multicore era, Computer 7 (2008) 33–38.
[23] I. Chung, G. Cong, D. Klepacki, S. Sbaraglia, S. Seelam, H.-F. Wen, et al., A framework for automated performance bottleneck detection, in: IPDPS 2008, IEEE International Symposium on Parallel and Distributed Processing, 2008, IEEE, 2008, pp. 1–7.

Rashid Al-Ali, PhD is the Deputy Chief Research Officer and Division Chief – Biomedical Informatics at Sidra Medical and Research Center where he will be establishing and leading a state-of-the-art Biomedical Informatics Division. Dr. Al-Ali is also responsible for setting up research groups in Bioinformatics, Clinical Informatics and High Performance Computing (HPC) for biomedical research and bioinformatics applications and analysis. Dr. Al Ali is also a Member of the IT Executive Committee at Sidra, which provides executive strategic decisions and recommendations for all aspects related to information technology. Prior to assuming his current role at Sidra, he served as Associate Director, e-Health & Medical Informatics from 2011 to May 2013. During 2011–2012, Dr. Al Ali also held the prestigious Harvard Fellowship with the Division of Clinical Informatics at the Harvard Medical Faculty Physicians at BIDMC in Boston. He took part in didactic lectures, formal Harvard MD/MBA courses participation, seminars and meetings with senior informaticians and CIOs on topics related to Health Information Technology. He also conducted scientific research investigations into health information exchange, electronic health and personal health records. Dr. Al-Ali joined Sidra from the ASPIRE Academy for Sports Excellence where he served as Chief Information Officer (CIO) beginning in 2007. At ASPIRE, he established state-of-the-art ICT infrastructure and was instrumental in automating business processes for business units. Dr. Al Ali also led the process of planning, designing and implementing a specialized software system called AIMS, unique to ASPIRE Academy, which models the athlete development life-cycle from talent identification to graduation from the academy. Dr. Al Ali also served as Director – Corporate Services at the ASPIRE Academy, where he oversaw the operation of information technology, finance and procurement, human resources, communication, general administration and deputizing the Director General, before joining Sidra. From 2005 to 2007, Dr. Al Ali served as Q-CERT Director, (Qatar Computer Emergency Response Team) in ictQATAR, (Supreme Council for Information and Communications Technology) where he was responsible for guidance and strategic direction of the Q-CERT program, building relationships with government and private organizations in Qatar and the Gulf Region and representing the state of Qatar in international ICT sector meetings, such as Gulf Cooperation Council/Arab League. Dr. Al-Ali received his Ph.D. in Computer Science from Cardiff University, Wales, UK in 2005 and his MS in Computer Science from The George Washington University, Washington DC, USA in 1997. He received his BS in Computer Engineering and Computer Science (with Honors) from the University of the Pacific, CA, USA in 1992. Dr. Al-Ali has a number of scientific publications in the area of Distributed Systems, Grid Computing and Clinical Informatics.

Nagarajan Kathiresan, PhD is the Research Scientist in High Performance Computing (HPC), Biomedical Informatics at Sidra Medical and Research Center where he will be optimizing and performance tuning of Bioinformatics application in the high-end HPC cluster. Dr. Nagarajan will be responsible for performance optimization of Next Generation Sequencing (NGS) Analysis, experimental NGS indexing, mapping & alignment of human genomes, sequence alignment & design/enhance using parallelism on multicore architectures. Additionally, he will be working on Parallel Data mining services like Map-Reduce, Task Parallel Library, MPICH synchronization protocols and many more. The NGS optimization & Data mining are part of our "Knowledge management in Translational medicine studies & Clinical Research". Previously, Dr. Nagarajan was working in HPC – Technical Computing team in Systems & Technology Group, India. He has over 8 years of experience in performance engineering & cluster optimization in applied HPC domains which includes quantum chemistry, molecular dynamics, weather modeling & life science applications. He also developed various tools, system libraries and dashboard tools for monitoring multi-process applications. He is interested to develop and enhance the application performance using parallelization, code vectorization and multicore aware algorithms on the multi-core cluster environment. He contributed "check-point & restart framework" for MPI Fault tolerance during his PhD research.

Mohammed El Anbari got his PhD in mathematics from the University of Paris-Sud 11 (France). Currently, he is a Biostatistics scientist at Sidra Medical and Research Center in Qatar. Before joining Sidra, he was a postdoc at the Qatar Computing Research Institute. He is a statistician with broad interest in statistical machine learning. He is developing new statistical methods to learn from high dimensional data especially from genomics. In his current job, he has also a special role providing Biostatistics expertise to multiple collaborative projects in Clinical, Basic, and Translational Biomedical Research.

Eric Schendel, PhD is Senior Software Architect at Sidra Medical and Research Center. Eric is involved in Innovative, results-driven researcher, software architect, and engineer with extensive experience and passion toward high performance computing (HPC). Proficient as a software designer, systems architect, and developer of enterprise-level frameworks to advance productivity and product optimizations of scientific, engineering, and manufacturing environments. Developed leadership abilities through coaching and mentoring while leading a team of researchers and senior software/hardware engineers. Established excellent interpersonal and communication skills through multi-team collaboration and acquiring buy-in from stakeholders for global product solutions. Areas of expertise include:

- New technology research/design/development.
- Parallel and distributed computing.
- Algorithm and HPC system optimization.
- Heterogeneous environment integration.

Tariq Abu Zaid, BE is the infrastructure manager at SIDRA Medical and Research, Doha, Qatar. He has Twenty years of interaction with the Information Technology through different industries, Government, Oil and Gas, Sports and Education. Along with the knowledge of sound IT standards and frameworks, created the capability to draw the technological enablement road map for the business and the maturity to build the technology architecture that promotes integration and reuse. I Conduct gap and impact analysis to plan achieving the future state in a safe and meaningful incremental phases considering IT and business maturity and capability to absorb the change and make it part of the operational phase. Employs IT Governance Knowledge to monitor the technology deployment life cycle from ideation, business case building, investment prioritization, risk mitigation, implementation, to value delivery and benefits realization and finally to retirement.